PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-126162
(43)Date of publication of application : 11.05.1999

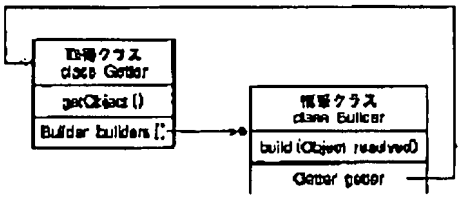
(51)Int.Cl. G06F 9/44
G06F 9/06

(21)Application number : 09-289690
(22)Date of filing : 22.10.1997

(71)Applicant : SONY CORP
(72)Inventor : NANBA SHINJI
MINEYAMA TORU

(54) APPARATUS AND METHOD FOR INFORMATION PROCESSING AND RECORDING MEDIUM

(57)Abstract:
PROBLEM TO BE SOLVED: To make an application program constructible by allowing an acquisition object to acquire an object to be constructed and allowing a constructing object that the acquisition object owns to initializes the object to be constructed that the acquisition object has acquired by a constructing method.
SOLUTION: The whole of a rectangle, which is divided longitudinally in three, represents a class and an acquisition class class Getter has an acquisition method getObject() and a Builder type field builders[]. Further, the constructing class class Builder has a constructing method build (Object resolved) and a Getter type field getter. Then, the acquisition class executes the acquisition method getObject() to obtain a specified object to be constructed and then return it as a return value. Further, the acquisition class owns a constructing object for initializing the object to be constructed that the acquisition object has obtained by executing the acquisition method getObject().



LEGAL STATUS

[Date of request for examination] 26.11.2003
[Date of sending the examiner's decision of rejection] 20.02.2006
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平11－126162

(43)公開日　平成11年(1999)５月11日

(51)Int.Cl.[6]		識別記号	FI		
G06F	9/44	530	G06F	9/44	530P
	9/06	530		9/06	530W

審査請求　未請求　請求項の数10　ＯＬ　（全 19 頁）

(21)出願番号　特願平9－289690

(22)出願日　平成９年(1997)10月22日

(71)出願人　000002185
ソニー株式会社
東京都品川区北品川６丁目７番35号

(72)発明者　難波　慎二
東京都品川区北品川６丁目７番35号　ソニー株式会社内

(72)発明者　峯山　徹
東京都品川区北品川６丁目７番35号　ソニー株式会社内

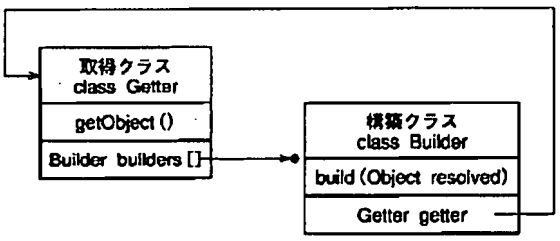
(74)代理人　弁理士　稲本　義雄

(54)【発明の名称】　情報処理装置および情報処理方法、並びに記録媒体

(57)【要約】

【課題】　アプリケーションプログラムなどの構築物を構成する被構築オブジェクトを集めて、その構築物を構築することができるようにする。

【解決手段】　構築クラスBuilderの構築オブジェクトは、被構築オブジェクトresolvedが有するフィールド設定メソッドをコールして、被構築オブジェクト間の参照関係を構築する構築メソッドbuild(Object resolved)を有している。取得クラスGetterの取得オブジェクトは、被構築オブジェクトを取得する取得メソッドgetObject()を有し、構築オブジェクトの所有が可能なようになされている。そして、取得オブジェクトにおいて、取得メソッドgetObject()によって被構築オブジェクトresolvedが取得され、その取得オブジェクトが所有する構築オブジェクトにおいて、構築メソッドbuild(Object resolved)によって、取得オブジェクトが取得した被構築オブジェクトresolvedが初期化されるようになされている。



1

【特許請求の範囲】

【請求項１】　所定の構築物を構成する複数の被構築オブジェクトを取得して、その初期化を行い、前記所定の構築物を構成するための処理を行う情報処理装置であって、

前記被構築オブジェクトが、自身のフィールドを設定することにより、その初期化を行うフィールド設定メソッドを有する場合に、そのフィールド設定メソッドをコールして、前記被構築オブジェクト間の参照関係を構築する構築メソッドを有する構築オブジェクトと、

前記被構築オブジェクトを取得する取得メソッドを有する、前記構築オブジェクトの所有が可能な取得オブジェクトとを備え、

前記取得オブジェクトは、前記取得メソッドによって前記被構築オブジェクトを取得し、

その取得オブジェクトが所有する前記構築オブジェクトは、前記構築メソッドによって、前記取得オブジェクトが取得した前記被構築オブジェクトを初期化することを特徴とする情報処理装置。

【請求項２】　前記構築オブジェクトは、前記取得オブジェクトの所有が可能であり、

前記被構築オブジェクトの初期化を行う前記構築オブジェクトが所有する前記取得オブジェクトは、その初期化に必要な前記被構築オブジェクトを取得することを特徴とする請求項１に記載の情報処理装置。

【請求項３】　前記取得オブジェクトが取得した前記被構築オブジェクトを登録するメソッドを有する登録オブジェクトをさらに備え、

前記取得オブジェクトは、初期化に必要な前記被構築オブジェクトが、前記登録オブジェクトによって既に登録されているとき、その既に登録されている被構築オブジェクトを取得することを特徴とする請求項２に記載の情報処理装置。

【請求項４】　前記被構築オブジェクトは、それとの新たな参照関係を構築する前記構築オブジェクトを所有可能なようになされていることを特徴とする請求項１に記載の情報処理装置。

【請求項５】　前記取得メソッドまたは構築メソッドは、ポリモーフィックなメソッドであることを特徴とする請求項１に記載の情報処理装置。

【請求項６】　前記被構築オブジェクトは、１以上の前記フィールド設定メソッドを有することが可能とされていることを特徴とする請求項１に記載の情報処理装置。

【請求項７】　前記取得オブジェクトは、１以上の前記構築オブジェクトを所有することが可能とされており、前記取得オブジェクトが所有する１以上の前記構築オブジェクトが、前記構築メソッドによって、前記取得オブジェクトが取得した前記被構築オブジェクトの１以上のフィールドを、それぞれ初期化することを特徴とする請求項６に記載の情報処理装置。

2

【請求項８】　前記複数の被構築オブジェクトが、場所的または時間的に分散していることを特徴とする請求項１に記載の情報処理装置。

【請求項９】　所定の構築物を構成する複数の被構築オブジェクトを取得して、その初期化を行い、前記所定の構築物を構成するための処理を行う情報処理装置の情報処理方法であって、

前記情報処理装置は、

前記被構築オブジェクトが、自身のフィールドを設定することにより、その初期化を行うフィールド設定メソッドを有する場合に、そのフィールド設定メソッドをコールして、前記被構築オブジェクト間の参照関係を構築する構築メソッドを有する構築オブジェクトと、

前記被構築オブジェクトを取得する取得メソッドを有する、前記構築オブジェクトの所有が可能な取得オブジェクトとを備え、

前記取得オブジェクトに、前記取得メソッドによって前記被構築オブジェクトを取得させ、

その取得オブジェクトが所有する前記構築オブジェクトに、前記構築メソッドによって、前記取得オブジェクトが取得した前記被構築オブジェクトを初期化させることを特徴とする情報処理方法。

【請求項１０】　コンピュータに実行させるコンピュータプログラムが記録されている記録媒体において、

所定の構築物を構成する複数の被構築オブジェクトを取得して、その初期化を行い、前記所定の構築物を構成する処理を、前記コンピュータに行わせるためのコンピュータプログラムであって、

前記被構築オブジェクトが、自身のフィールドを設定することにより、その初期化を行うフィールド設定メソッドを有する場合に、そのフィールド設定メソッドをコールして、前記被構築オブジェクト間の参照関係を構築する構築メソッドを有する構築オブジェクト、および前記被構築オブジェクトを取得する取得メソッドを有する、前記構築オブジェクトの所有が可能な取得オブジェクトを含み、

前記取得オブジェクトに、前記取得メソッドによって前記被構築オブジェクトを取得させ、

その取得オブジェクトが所有する前記構築オブジェクトに、前記構築メソッドによって、前記取得オブジェクトが取得した前記被構築オブジェクトを初期化させる処理を行わせるためのコンピュータプログラムが記録されていることを特徴とする記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、情報処理装置および情報処理方法、並びに記録媒体に関し、特に、例えば、アプリケーションプログラムを構成する複数のオブジェクトや、マルチメディアコンテンツその他のデータを構成する複数のオブジェクトが分散していても、それ

らのオブジェクトを取得してリンクさせることにより、アプリケーションプログラムやマルチメディアコンテンツその他のデータを構成（構築）することができるようにする情報処理装置および情報処理方法、並びに記録媒体に関する。

【０００２】

【従来の技術】近年、オブジェクト指向技術の分野では、コンポーネントソフトウェア技術が注目されている。ここで、コンポーネントソフトウェア技術とは、アプリケーションプログラムや、アプリケーションプログラムで利用されるマルチメディアコンテンツその他のデータなどを、複数のコンポーネント（部品）の結合により構築する手法（技術）である。

【０００３】

【発明が解決しようとする課題】ところで、コンポーネントソフトウェア技術を利用して構築されたアプリケーションプログラムを構成する複数のオブジェクトは、それらの参照関係を、そのアプリケーションプログラムを構成するクラスに静的にリンクさせた形で提供されるようになされていた。

【０００４】また、アプリケーションプログラムが利用するマルチメディアコンテンツやドキュメントのデータなどを構成する複数のオブジェクトは、バイト列に、いわばシリアライズされて、ファイルに書き込まれた状態で提供されるようになされていた。

【０００５】従って、そのような複数のオブジェクトを、例えば、ネットワーク上に分散させておき、必要なときに、必要なオブジェクトだけを集めて、動的にリンクさせ（参照関係を構築し）、アプリケーションプログラムやマルチメディアコンテンツなどを構成することが困難であった。

【０００６】本発明は、このような状況に鑑みてなされたものであり、例えば、アプリケーションプログラムを構成する複数のオブジェクトやマルチメディアコンテンツを構成する複数のオブジェクトが分散していても、それらのオブジェクトを取得して、動的にリンクさせることにより、アプリケーションプログラムやマルチメディアコンテンツを構成（構築）することができるようにするものである。

【０００７】

【課題を解決するための手段】請求項１に記載の情報処理装置は、被構築オブジェクトが、自身のフィールドを設定することにより、その初期化を行うフィールド設定メソッドを有する場合に、そのフィールド設定メソッドをコールして、被構築オブジェクト間の参照関係を構築する構築メソッドを有する構築オブジェクトと、被構築オブジェクトを取得する取得メソッドを有する、構築オブジェクトの所有が可能な取得オブジェクトとを備え、取得オブジェクトが、取得メソッドによって被構築オブジェクトを取得し、その取得オブジェクトが所有する構築オブジェクトが、構築メソッドによって、取得オブジェクトが取得した被構築オブジェクトを初期化することを特徴とする。

【０００８】請求項９に記載の情報処理方法は、取得オブジェクトに、取得メソッドによって被構築オブジェクトを取得させ、その取得オブジェクトが所有する構築オブジェクトに、構築メソッドによって、取得オブジェクトが取得した被構築オブジェクトを初期化させることを特徴とする。

【０００９】請求項１０に記載の記録媒体は、被構築オブジェクトが、自身のフィールドを設定することにより、その初期化を行うフィールド設定メソッドを有する場合に、そのフィールド設定メソッドをコールして、被構築オブジェクト間の参照関係を構築する構築メソッドを有する構築オブジェクト、および被構築オブジェクトを取得する取得メソッドを有する、構築オブジェクトの所有が可能な取得オブジェクトを含み、取得オブジェクトに、取得メソッドによって被構築オブジェクトを取得させ、その取得オブジェクトが所有する構築オブジェクトに、構築メソッドによって、取得オブジェクトが取得した被構築オブジェクトを初期化させる処理を、コンピュータに行わせるためのコンピュータプログラムが記録されていることを特徴とする。

【００１０】請求項１に記載の情報処理装置においては、構築オブジェクトは、被構築オブジェクトが有するフィールド設定メソッドをコールして、被構築オブジェクト間の参照関係を構築する構築メソッドを有している。取得オブジェクトは、被構築オブジェクトを取得する取得メソッドを有し、構築オブジェクトの所有が可能なようになされている。この場合において、取得オブジェクトが、取得メソッドによって被構築オブジェクトを取得し、その取得オブジェクトが所有する構築オブジェクトが、構築メソッドによって、取得オブジェクトが取得した被構築オブジェクトを初期化するようになされている。

【００１１】請求項９に記載の情報処理方法においては、取得オブジェクトに、取得メソッドによって被構築オブジェクトを取得させ、その取得オブジェクトが所有する構築オブジェクトに、構築メソッドによって、取得オブジェクトが取得した被構築オブジェクトを初期化させるようになされている。

【００１２】請求項１０に記載の記録媒体には、被構築オブジェクトが、自身のフィールドを設定することにより、その初期化を行うフィールド設定メソッドを有する場合に、そのフィールド設定メソッドをコールして、被構築オブジェクト間の参照関係を構築する構築メソッドを有する構築オブジェクト、および被構築オブジェクトを取得する取得メソッドを有する、構築オブジェクトの所有が可能な取得オブジェクトを含み、取得オブジェクトに、取得メソッドによって被構築オブジェクトを取得

5

させ、その取得オブジェクトが所有する構築オブジェクトに、構築メソッドによって、取得オブジェクトが取得した被構築オブジェクトを初期化させる処理を、コンピュータに行わせるためのコンピュータプログラムが記録されている。

【００１３】

【発明の実施の形態】図１は、本発明を適用した放送システム（システムとは、複数の装置が論理的に集合した物をいい、各構成の装置が同一筐体中にあるか否かは問わない）の一実施の形態の構成例を示している。

【００１４】放送局１においては、テレビジョン放送信号としての電波が送信され、この電波は、衛星（通信衛星または放送衛星）２を介して、アンテナ３で受信される。アンテナ３で受信された受信信号は、ＳＴＢ（Set Top Box）４に供給され、所定のチャンネルのテレビジョン放送信号が取り出される。このテレビジョン放送信号は、ＳＴＢ４からＣＲＴ５に供給されて表示される。

【００１５】なお、ＳＴＢ４では、放送局１以外の図示せぬ放送局からのテレビジョン放送信号も受信することができるようになされている。

【００１６】ところで、ＳＴＢ４では、視聴者が、各チャンネルにおいて放送されている番組の概要を容易に認識することができるように、ＥＰＧ（Electric Program Guide）を、ＣＲＴ５に表示させることができるようになされている。

【００１７】しかしながら、紹介するチャンネル数が多く、さらに、番組の紹介を、その番組名やタイトルなどのテキストの表示だけでなく、動画や音声などを用いて行う場合、即ち、マルチメディアコンテンツとしてのＥＰＧを用いる場合には、そのデータ量が膨大なものとなり、これを、衛星２を介して、ＳＴＢ４に送信するのでは、通常の番組放送に支障をきたすことがある。また、ＥＰＧの内容を動的に変更したい場合があるが、放送局１において、ＥＰＧを構成して送信するのでは、その動的な変更は困難である。

【００１８】そこで、ここでは、ＥＰＧを、それを構成する各種の部品に分けて分散させておき、ＳＴＢ４において、それらの部品を集めて、ＥＰＧを、動的に構成することができるようになされている。

【００１９】即ち、図１の実施の形態では、ＥＰＧを構成する部品が、例えば、インターネット６上に分散している。具体的には、ここでは、インターネット６に、放送局１，ＳＴＢ４、サーバ７が接続されており、それぞれに、ＥＰＧを構成する部品が記憶されている。そして、ＳＴＢ４では、自身が記憶している部品の他、放送局１やサーバ７に記憶されている部品が、インターネット６を介して、必要に応じて集められ、ＥＰＧが構成されるようになされている。

【００２０】次に、図２は、図１のＳＴＢ４の構成例を示している。

6

【００２１】通信Ｉ／Ｆ（Interface）１１は、例えば、モデムなどでなり、インターネット６との間の通信制御を行うようになされている。受信処理部１２は、例えば、ダウンコンバータや、チューナ、復調回路、さらにはデスクランブラなどで構成され、アンテナ３からの受信信号から、所定のテレビジョン放送信号を取り出し、必要に応じてデスクランブルして、表示制御部１６に供給するようになされている。

【００２２】ＲＯＭ（Read Only memory）１３は、例えば、ＩＰＬ（Initial Program Loading）のプログラムなどを記憶している。ＣＰＵ（Central Processing Unit）１４は、ハードディスク１７に記憶されているＯＳ（Operating System）の制御の下、同じくハードディスク１７に記憶されている各種のアプリケーションプログラムを実行することで、ＥＰＧの部品を集めてリンクさせ、ＥＰＧを構成する処理（この処理を、以下、適宜、構築処理といい、また、構築処理を行うためのアプリケーションプログラムを、以下、適宜、構築プログラムという）などを行うようになされている。ＲＡＭ（Random Access memory）１５は、ＣＰＵ１４が実行するプログラムや、その動作上必要なデータなどを一時記憶するようになされている。表示制御部１６は、そこに供給されるデータを、ＣＲＴ５に表示させるための表示制御を行うようになされている。

【００２３】ハードディスク１７は、ＯＳやアプリケーションプログラム、さらには、必要に応じて、ＥＰＧを構成する部品を記憶している。なお、ＥＰＧを構成する部品は、例えば、通信Ｉ／Ｆ１１に、インターネット６を介して放送局１やサーバ７と通信を行わせることにより入手することができる。また、例えば、放送局１に、ＥＰＧを構成する部品を放送させ、これを受信処理部１２において受信することなどによっても入手することができる。また、ハードディスク１７は、ＣＰＵ１４が処理を行うのに必要なデータや、処理を行った結果得られるデータなども、必要に応じて記憶するようになされている。

【００２４】ＣＤ－ＲＯＭ（Compact Disc ROM）１８には、例えば、ＥＰＧを構成する部品などが記憶されている。操作部１９は、例えば、キーボードや、マウス、その他の各種のボタンなどで構成され、データやコマンドを入力するときに操作される。

【００２５】なお、以上の各ブロックは、バスを介して相互に接続されている。

【００２６】以上のように構成されるＳＴＢ４では、例えば、操作部１９が、所定のチャンネルを受信するように操作されると、受信処理部１２において、アンテナ３からの受信信号から、その操作に対応したチャンネルのテレビジョン放送信号が抽出され、表示制御部１６を介してＣＲＴ５に供給されて表示される。

【００２７】また、例えば、操作部１９が、ＥＰＧを表

示するように操作されると、ＣＰＵ１４では、ハードディスク１７に記憶された構築プログラムが実行されることで、構築処理が行われ、これにより、ＥＰＧが構成される。即ち、ＥＰＧを構成する部品が、ハードディスク１７やＣＤ－ＲＯＭ１８から読み出されたり、通信Ｉ／Ｆ１１において、インターネット６を介して、放送局１やサーバ７と通信が行われることにより取得される。そして、ＣＰＵ１４は、その取得した部品どうしを、動的にリンクさせ、ＥＰＧを構成する。このＥＰＧは、表示制御部１６を介してＣＲＴ５に供給されて表示される。

【００２８】なお、ここでは、構築プログラムは、オブジェクト指向言語の１つである、例えば、Java言語で記述されている（Javaは商標）。また、それに対応して、以下に示すプログラムリストは、Java言語で記述してある。なお、Java言語の詳細については、例えば、「Java APIプログラミングガイド」、青柳龍也、工学図書や、「Javaパワープログラミング」、Paul Tyma/Gabriel Torok/Troy Dowing、Soft Bank、「The Java Language Specification」、Java Soft、「Java Core Reflecion, API and Specification」、Java Softなどに開示されているので、これらを参照されたい。

【００２９】次に、図３は、構築処理によって構成されるＥＰＧの例を示している。この実施の形態では、ＥＰＧは、例えば、ニュース番組についてのものとなっており、８つの部品Ｐ１乃至Ｐ８から構成されている。

【００３０】部品Ｐ１は、ＥＰＧの背景（バックグラウンド）で、他の部品Ｐ２乃至Ｐ４およびＰ５が張り付けられている。部品Ｐ２乃至Ｐ４は、ニュースのジャンルを選択するためのボタンで、ここでは、部品Ｐ２乃至Ｐ４を操作することにより、部品Ｐ６乃至Ｐ８が変化するようになされている。即ち、ボタンとしての部品Ｐ２乃至Ｐ４が操作された場合（例えば、マウスでクリックされた場合）、部品Ｐ６乃至Ｐ８は、例えば、政治、国際、社会それぞれについてのニュースに関係する内容に変化するようになされている。部品Ｐ５は、部品Ｐ６乃至Ｐ８が張り付けられる背景を構成している。部品Ｐ６乃至Ｐ８は、テキストデータ、動画データ、静止画データで、それぞれには、ボタンとしての部品Ｐ２乃至Ｐ４の操作に対応したジャンル内容がテキスト、動画、静止画で表示されるようになされている。

【００３１】なお、例えば、部品Ｐ６乃至Ｐ８に、あるジャンルのニュースについての情報が表示されている場合において、そのうちのいずれかがクリックされると、そのジャンルのニュース番組が、図３のＥＰＧに代えて表示されるようになされている。

【００３２】いま、ＥＰＧを構成する部品Ｐ１乃至Ｐ８は、オブジェクト（被構築オブジェクト）ｏｂｊ１乃至ｏｂｊ８でそれぞれ構成され、ここでは、例えば、図４に示すように、ネットワーク上に分散しているとする。即ち、ここでは、被構築オブジェクトｏｂｊ１およびｏｂｊ２はサーバＳＶ１上に、被構築オブジェクトｏｂｊ３乃至ｏｂｊ５はサーバＳＶ２上に、被構築オブジェクトｏｂｊ６はサーバＳＶ３上に、被構築オブジェクトｏｂｊ７およびｏｂｊ８はユーザ端末ＵＴ上に、それぞれ記憶されている。また、被構築オブジェクトｏｂｊ１乃至ｏｂｊ８は、互いに、静的にリンクされていない状態（いわば独立の状態）で記憶されている。

【００３３】なお、図４におけるサーバＳＶ１乃至ＳＶ３は、図１におけるインターネット６上のサーバ（例えば、サーバ７など）に相当し、図４におけるユーザ端末ＵＴは、図１におけるＳＴＢ４に相当する。

【００３４】この場合において、ユーザ端末ＵＴ上では、構築プログラムが実行されることにより、ネットワーク上に分散している被構築オブジェクトｏｂｊ１乃至ｏｂｊ８が取得されて、リンク（参照関係）が形成され、これにより、構築物としてのＥＰＧが構成（構築）される。

【００３５】このため、構築プログラムでは、被構築オブジェクトが、自身のフィールドを設定することにより、その初期化を行うフィールド設定メソッドを有する場合に、そのフィールド設定メソッドをコールして、被構築オブジェクト間の参照関係を構築する構築メソッドを有する構築クラス、被構築オブジェクトを取得する取得メソッドを有する、構築オブジェクトの所有が可能な取得クラス、および取得オブジェクトが取得した被構築オブジェクトを登録するメソッドを有する登録クラスが定義されている。

【００３６】即ち、取得クラスは、ネットワーク上に分散する被構築オブジェクトを取得するための取得メソッドgetObject()を有し、その取得メソッドgetObject()を実行することにより、所定の被構築オブジェクトを取得して、戻り値として返すようになされている。さらに、取得クラスにおいては、そのインスタンスである取得オブジェクトが、取得メソッドgetObject()を実行することにより取得した被構築オブジェクトの初期化を行うための、構築オブジェクトのインスタンスである構築オブジェクトを所有することができるように、定義がなされている。即ち、取得クラスでは、構築クラスの型を、例えば、Builderと表すと、Builder型のフィールドが定義されている。

【００３７】構築クラスは、被構築オブジェクト間の参照関係の構築、即ち、被構築オブジェクトの初期化を行うための構築メソッドbuild(Object resolved)を有している。ここで、構築メソッドbuild(Object resolved)の引数である、Object型のresolvedは、初期化対象の被構築オブジェクトを表している。また、その型であるObjectは、Javaにおいて、最上位階層のクラスとして定義されているクラスである。

【００３８】構築クラスにおいては、そのインスタンスである構築オブジェクトが、構築メソッドbuild(Object

resolved)を実行するのに必要な被構築オブジェクトを取得するための取得オブジェクトを所有することができるように、定義がなされている。即ち、例えば、ある被構築オブジェクトresolvedAを取得した後、その被構築オブジェクトresolvedAから、他の被構築オブジェクトresolvedBに対して参照関係がある場合には、被構築オブジェクトresolvedAの初期化には、被構築オブジェクトresolvedBの参照が必要であり、従って、被構築オブジェクトresolvedBを取得する必要がある。このため、構築クラスでは、取得クラスの型を、例えば、Getterと表すと、Getter型のフィールドが定義されている。

【００３９】従って、取得クラスと構築クラスとは、図５に示すような関係を有する。なお、図５においては、縦方向に３分割された長方形全体がクラスを表している。また、図６（Ａ）に示すように、クラスを表す長方形の最も上の段には、クラス名が、上から２番目の段には、そのクラスで定義されているメソッド名が、最も下の段には、そのクラスで定義されているフィールド名が、それぞれ配置されている。

【００４０】上述したように、取得クラスGetterは、取得メソッドgetObject()と、Builder型のフィールドbuilders[]を有している。ここで、Builder型のフィールドが配列になっているが、これについては後述する。

【００４１】また、構築クラスBuilderは、構築メソッドbuild(Object resolved)と、Getter型のフィールドgetterを有している。

【００４２】なお、図５における矢印は、その矢印の始端のクラス（ここでは、取得クラスまたは構築クラス）のインスタンスが、その終端のクラス（ここでは、構築クラスまたは取得クラス）の型のフィールド（その終端のクラスのインスタンス）を所有可能なことを意味している。また、図６（Ｂ）に示すように、その終端に●印がある矢印は、その始端のクラスのインスタンスが、その終端のクラスのインスタンスを、複数所有可能であることを表し、●印のない矢印は、その始端のクラスのインスタンスが、その終端のクラスのインスタンスを、１だけ所有可能であることを表している。従って、ここでは、取得オブジェクトは、複数の構築オブジェクトを所有可能であるが、構築オブジェクトは、１の取得オブジェクトだけを所有可能なようになされている。

【００４３】次に、取得クラスにおいて定義されている取得メソッドgetObject()は、ここでは、いわゆるポリモーフィック（polymorphic：多様的）なメソッド（例えば、あるスーパクラスにおいて定義されているメソッドであって、その２以上のサブクラスそれぞれにおいて実装が与えられており、いずれの実装を有するメソッドとしても利用可能なもの）とされており、これにより、そのサブクラスにおいて、種々の実装を与えて、所望の実装が与えられたものを利用することができるようになされている。

【００４４】即ち、ここでは、取得メソッドgetObject()は、取得クラスではなく、そのサブクラスにおいて実装が与えられている。具体的には、例えば、図７に示すように、取得クラスのサブクラスとして、ネットワーク取得クラスNetGetter、ファイル取得クラスFileGetter、オブジェクト保持取得クラスSaveGetter、スタブ取得クラスStubGetter、および生成取得クラスConstructGetterの５つが定義されている。

【００４５】ネットワーク取得クラスNetGetterでは、リモートのホストコンピュータとしての、例えば、サーバＳＶ１乃至ＳＶ３においてシリアライズされてファイルに書き込まれている被構築オブジェクトをロードし、そのデシリアライズ（シリアライズされてバイト列にされている被構築オブジェクトを使用可能な状態にすること）を行う取得メソッドgetObject()が実装されている。また、ネットワーク取得クラスNetGetterには、シリアライズされてファイルに書き込まれている被構築オブジェクトを、そこに実装されている取得メソッドgetObject()に取得させるために必要な、その被構築オブジェクトのＵＲＬ（Uniform Resource Locator）が格納されるフィールド（URL型のフィールド）urlも定義されている。

【００４６】ファイル取得クラスFileGetterでは、ユーザ端末ＵＴにおいてシリアライズされてファイル（ローカルファイル）に書き込まれている被構築オブジェクトをロードし、そのデシリアライズを行う取得メソッドgetObject()が実装されている。また、ファイル取得クラスFileGetterには、シリアライズされてファイルに書き込まれている被構築オブジェクトを、そこに実装されている取得メソッドgetObject()に取得させるために必要な、そのファイルのファイル名を与えるフィールド（String型のフィールド）fileNameも定義されている。

【００４７】オブジェクト保持取得クラスSaveGetterは、被構築オブジェクトを所有しており、そこでは、その被構築オブジェクトを取得する取得メソッドgetObject()が実装されている。また、オブジェクト保持取得クラスSaveGetterには、自身で所有している被構築オブジェクトを、そこに実装されている取得メソッドGetObject()に取得させるために必要な、その被構築オブジェクトを識別するための値を与えるフィールド（Object型のフィールド）valueも定義されている。

【００４８】スタブ取得クラスStubGetterでは、リモートのホストコンピュータとしての、例えば、サーバＳＶ１乃至ＳＶ３において、スタブとして提供される被構築オブジェクトを取得する取得メソッドgetObject()が実装されている。また、スタブ取得クラスStubGetterには、被構築オブジェクトを、スタブとして提供するサーバから、その被構築オブジェクトを、そこに実装されている取得メソッドgetObject()に取得させるために必要な、そのサーバのＩＰ（Internet Protocol）アドレス

11

が格納されるフィールド (IPAddress server long型のフィールド) objectIDも定義されている。

【００４９】生成取得クラスConstructGetterでは、所定のクラスのインスタンスである被構築オブジェクトを、そのコンストラクタをコールすることにより生成させて取得する取得メソッドgetObject()が実装されている。また、生成取得クラスConstructGetterには、コンストラクタをコールすることにより生成される被構築オブジェクトを、そこに実装されている取得メソッドgetObject()に取得させるために必要な、その被構築オブジェクトのクラス名が格納されるフィールド (String型のフィールド) classNameも定義されている。

【００５０】以上のように、取得メソッドgetObject()をポリモーフィックなメソッドとし、取得クラスのサブクラスにおいて、取得メソッドgetObject()の実装を与えるようにすることで、各種の被構築オブジェクトを、それぞれの取得方法に対応した取得クラスを定義しなくても、対応するサブクラスにおける取得メソッドgetObject()をコールすることで取得することが可能となる。

12

【００５１】なお、取得クラスのサブクラスは、上述したものに限定されるものではなく、必要な取得メソッドgetObject()を実装した種々のサブクラスを定義することが可能である。

【００５２】以上のような取得クラスのインスタンスである取得オブジェクトでは、それが有する取得メソッドgetObject()によって、対応する被構築オブジェクトが取得される。このようにして取得された被構築オブジェクトは、そこに定義されているフィールドに、他の被構築オブジェクトの参照を設定することにより、その初期化を行う必要がある場合がある。即ち、例えば、ある取得オブジェクトによって、次のように定義されるクラスResolvedCのインスタンスとしての被構築オブジェクトresolvedCが取得されたとする。

【００５３】

```
class ResolvedC extends Object{
        private int fieldA;
        private int fieldB;
        public void setFieldA(int f){
                fieldA=f;
        }
        public void setFieldB(int f){
                fieldB=f;
        }
        ...
}
```

・・・（1）

【００５４】ここで、クラスResolvedCは、クラスObjectをスーパクラスとするサブクラスとされている (Javaでは、いかなるオブジェクトも、最終的には、オブジェクトクラスObjectに属する)。また、クラスResolvedCには、フィールドfieldA, fieldBが定義されており、さらに、それぞれのフィールドを設定することにより、その初期化を行うためのフィールド設定メソッドsetFieldA(int f), setFieldB(int f)も定義されている。なお、本実施の形態では、フィールド設定メソッドは、公開されているものとする。

【００５５】この場合、被構築オブジェクトresolvedCを取得した取得オブジェクトは、その被構築オブジェクトresolvedCを初期化するための構築オブジェクトを所有しており、即ち、フィールド設定メソッドsetFieldA(int f), setFieldB(int f)をコールする構築メソッドを有する構築オブジェクトを所有しており、被構築オブジェクトresolvedCの取得後は、その構築オブジェクトが有する構築メソッドがコールされる。これにより、その構築メソッドによって、フィールド設定メソッドsetFieldA(int f), setFieldB(int f)がコールされ、その結果、フィールドfieldA, fieldBが設定される。

【００５６】なお、プログラムリスト（1）に示したクラスResolvedCの被構築オブジェクトresolvedCは、設定が必要なフィールドとして、２つのフィールドfieldA, fieldBを有するが、このように、被構築オブジェクトが、複数の初期化すべきフィールドを有する場合には、取得オブジェクトに所有させる構築オブジェクトを、配列で宣言しておくと、その初期化を、容易に行うことが可能となる。

【００５７】即ち、被構築オブジェクトresolvedCを取得する取得オブジェクトにおいて、フィールド設定メソッドsetFieldA(int f), setFieldB(int f)をコールする構築メソッドを有する構築オブジェクトを、Builder型の配列builders[]として宣言する。この場合、その取得オブジェクトに、例えば、次のようなプログラムを記述しておくことで、取得した被構築オブジェクトが有する複数のフィールドすべての初期化を行うことができる。

【００５８】

```
for (int j=0; j<builders.leng
th; j++)
```

13　　14

```
builders[j].build(resolved);
```

・・・(2)

ここで、ｂｕｉｌｄｅｒｓ．ｌｅｎｇｔｈは、配列であるフィールドbuilders[]の長さを表す。また、構築メソッドbuild(resolved)の引数resolvedは、上述したように、初期化対象の被構築オブジェクトの参照、即ち、ここでは、被構築オブジェクトresolvedCの参照を表す。

【００５９】なお、被構築オブジェクトの初期化が必要ない場合（フィールドを有していない場合や、有していても、そのフィールドの初期化が必要でない場合）には、その被構築オブジェクトを取得する取得オブジェクトには、初期化のための構築オブジェクトを所有させる必要はない。

【００６０】また、例えば、被構築オブジェクトを取得する取得オブジェクトに、その被構築オブジェクトが有するフィールドの数と同一の数の構築オブジェクトを所有させる必要は、必ずしもない。即ち、被構築オブジェクトが有するフィールドのうちの１以上が、その取得時に、何らかの理由により、既に初期化されている場合には、その被構築オブジェクトを取得する取得オブジェクトには、初期化されていないフィールドを初期化するための構築オブジェクトだけを所有させれば充分である。

【００６１】さらに、被構築オブジェクトが、複数の初期化すべきフィールドを有する場合には、上述したように、取得オブジェクトに所有させる構築オブジェクトを、配列で宣言しておく他、例えば、いわゆるコンテナクラスを、構築クラスのサブクラスとしておき、そのコンテナクラスを利用することによっても、複数のフィールドの初期化を、容易に行うことが可能となる。

【００６２】また、プログラムリスト（２）においては、for文を用いるようにしたが、その他、例えば、while文などを用いることも可能である。

【００６３】次に、例えば、ある取得オブジェクトによって、次のように定義されるクラスResolvedAのインスタンスとしての被構築オブジェクトresolvedAが取得されたとする。

【００６４】

```
class ResolvedA extends Object{
        private ResolvedB fieldA;
        private Object painters[];
        public void setFieldA(ResolvedB f){
                fieldA=f;
        }
        public void addPainter(Object p,int index){
                painters[index]=p;
        }
        ...
}
```

・・・(3)

【００６５】ここで、クラスResolvedAは、クラスObjectをスーパクラスとするサブクラスとされている。また、クラスResolvedAには、フィールドとして、ResolvedB型のfieldAと、Object型のpainters[]が定義されており、さらに、それぞれのフィールドを設定することにより、その初期化を行うためのフィールド設定メソッドsetFieldA(ResolvedB f), addPainter(Object p,int index)も定義されている。

【００６６】なお、ResolvedBは、Objectクラスのサブクラスとして定義されているものとする。

【００６７】プログラムリスト（３）に示す（クラスの）被構築オブジェクトresolvedAを取得した取得オブジェクトは、その初期化をするための構築オブジェクト、即ち、まず、フィールドfieldAを設定するためのフィールド設定メソッドsetFieldA(ResolvedB f)をコールする構築メソッドを有する構築オブジェクトを有しており、その構築オブジェクトの構築メソッドbuild(resolvedA)がコールされることにより、フィールドfieldAが初期化される。

【００６８】しかしながら、この場合、フィールドfieldAを初期化するには、クラスResolvedBのインスタンスである被構築オブジェクトresolvedBの参照が必要である。

【００６９】そこで、フィールド設定メソッドsetFieldA(ResolvedB f)をコールする構築メソッドを有する構築オブジェクトは、被構築オブジェクトresolvedBを取得する取得メソッドgetObject()を有する、取得クラスの取得オブジェクトを所有している。

【００７０】そして、フィールド設定メソッドsetFieldA(ResolvedB f)をコールする構築メソッドを有する構築オブジェクトにおいて、それが所有する取得オブジェクトの取得メソッドgetObject()がコールされることにより、被構築オブジェクトresolvedBが取得され、これにより、その被構築オブジェクトresolvedBの参照を用

い、フィールド設定メソッドsetFieldA(ResolvedB f)によって、フィールドfieldAが初期化される。

【００７１】なお、被構築オブジェクトresolvedBの初期化が必要な場合は、被構築オブジェクトresolvedBを取得した取得オブジェクトが、その初期化のための構築オブジェクトを所有しており、その構築オブジェクトによって、被構築オブジェクトresolvedBの初期化が行われる。即ち、ある被構築オブジェクトの取得後、その初期化に必要な被構築オブジェクトがある場合には、その被構築オブジェクトの取得が行われ、以下、同様に、初期化に必要な被構築オブジェクトの取得が行われる。そして、初期化に必要な被構築オブジェクトすべてが取得されると、各被構築オブジェクトの初期化が行われていく（この点については、さらに後述する）。

【００７２】ところで、構築クラスにおいて定義されている構築メソッドbuild(Object resolved)を、その構築クラスに実装させると、初期化の処理が異なる被構築オブジェクトごとに、構築クラスを用意する必要が生じる。そこで、上述の取得クラスの取得メソッドgetObject()における場合と同様に、構築クラスにおいて定義されている構築メソッドbuild(Object resolved)も、ポリモーフィックなメソッドとし、そのサブクラスにおいて、種々の実装を与えて、所望の実装が与えられたものを利用するようにすることができる。ここでは、例えば、図８に示すように、構築クラスBuilderのサブクラスとして、セットメソッド構築クラスSetBuilderおよびアドメソッド構築クラスAddBuilderの２つが定義されている。

【００７３】即ち、例えば、プログラムリスト（３）に示したフィールド設定メソッドsetFieldA(ResolvedB f)によって、フィールドfieldAを初期化するためには、例えば、次のようになクラスSetBuilderを、構築クラスBuilderのサブクラスとして定義しておく。

【００７４】

```
class SetBuilder extends Builder{
        String fieldName;
        String methodName;
        Getter getter;
        void build(Object resolved){
                Method method=getSetMethod(fieldName, methodName, resolved
);
                Object objs[]={getter.getObject()};
                method.invoke(resolved, objs);
        }
}
```

・・・（４）

【００７５】ここで、メソッドgetSetMethod(fieldName, methodName, resolved)は、fieldNameで表されるフィールドの初期化を行う、methodNameで表されるフィールド設定メソッドを有するオブジェクトを生成するもので、プログラムリスト（４）では、そのオブジェクトが、Method型のフィールドmethodに設定されるようになされている（Methodクラスのインタンスmethodが生成されるようになされている）。なお、メソッドgetSetMethod(fieldName, methodName, resolved)では、fieldNameおよびresolvedから、methodNameで表されるフィールド設定メソッドを有するオブジェクトの型を認識し、それに対応するクラス（被構築オブジェクトのクラス）を探すことで、methodNameで表されるフィールド設定メソッドを見つけ出すようになされている。

【００７６】また、プログラムリスト（４）では、Object objs[]={getter.getObject()};において、Object型の配列（フィールド）objs[]に、取得オブジェクトgetterが有する取得メソッドgetObject()が取得する被構築オブジェクトが１つだけ、配列の要素として設定されるようになされている。

【００７７】さらに、クラスMethodでは、resolvedで表される被構築オブジェクトを、objsで表される被構築オブジェクトで初期化するメソッドinvoke（resolved, objs)が定義されており、プログラムリスト（４）では、method.invoke(resolved, objs);において、オブジェクトmethodにおけるメソッドinvoke(resolved, objs)をコールすることで、被構築オブジェクトresolvedを、被構築オブジェクトobjsの参照で初期化する、オブジェクトmethodが有するフィールド設定メソッド（ここでは、methodNameで表されるメソッド）がコールされるようになされている。なお、メソッドinvoke(resolved, objs)の第２引数（２番目の引数）は、配列とすることが規定されているため、objsは配列型のフィールドにされている。

【００７８】プログラムリスト（４）に示したように、サブクラスSetBuilderにおいて、取得メソッドbuild(Object resolved)を、リフレクション（クラスなどの定義構造を言語体系に反映させること）を使用して実装することで、即ち、初期化すべきフィールドのフィールド名fieldNameと、その初期化を行うためのフィールド設定メソッドのメソッド名methodNameを与えることによって、オブジェクトｍｅｔｈｏｄが有するフィールド設定メソッド（これは、取得メソッドｇｅｔＯｂｊｅｃ

t ()によって取得される被構築オブジェクトのフィールド設定メソッドに等しい）がコールされるようにすることで、種々の被構築オブジェクトの初期化が可能となる。なお、プログラムリスト（４）におけるfieldNameおよびmethodNameは、例えば、コンストラクタによって与えるようにすることができる。

【００７９】また、例えば、プログラムリスト（３）に

```
class AddBuilder extends Builder{
        String  fieldName;
        String  methodName;
        int     index;
        Getter  getter;
        void build(Object resolved){
                Method mehtod=getAddMethod(fieldName,methodName,resolove
d);
                Object objs[]={getter.getObject(),new Integer(index)};
                method.invoke(resolved,objs);
        }
}
```

・・・（５）

【００８１】ここで、メソッドgetAddMethod(fieldName,methodName,resolved)は、プログラムリスト（４）におけるメソッドgetSetMethod(fieldName,methodName,resolved)と同様に、fieldNameで表されるフィールドの初期化を行う、methodNameで表されるフィールド設定メソッドを有するオブジェクトを生成するものであるが、fieldNameから、methodNameで表されるフィールド設定メソッドを有するオブジェクトの型が、何らかのクラスの配列であることを認識し、さらに、その配列のクラスを認識してから、methodNameで表されるフィールド設定メソッドを見つけ出す点が、メソッドgetSetMethod(fieldName,methodName,resolved)と異なる。なお、ここでは、methodNameで表されるフィールド設定メソッドは、例えば、プログラムリスト（３）に示したaddPainter(Object p,int index)のように、第２引数にint型のフィールドをとるものと、あらかじめ決められており、これにより、上述の配列のクラスを認識することができるようになされている（ここでは、第２引数にint型のフィールドをとるものと認識される）。

【００８２】プログラムリスト（５）では、Object型の配列（フィールド）objs[]に、取得オブジェクトgetterが有する取得メソッドgetObject()が取得する被構築オブジェクトと、コンストラクタInteger(index)によって生成されるインスタンス（indexの値に対応する、Integer型のインスタンス）との２つが、配列の要素として設定されるようになされている。

【００８３】そして、プログラムリスト（５）でも、プログラムリスト（４）における場合と同様に、オブジェクトmethodにおけるメソッドinvoke(resolved,objs)をコールすることで、被構築オブジェクトresolvedを、被構築オブジェクトobjsの参照で初期化する、オブジェクトmethodが有するフィールド設定メソッド（ここでは、methodNameで表されるメソッド）がコールされるようになされている。

【００８４】プログラムリスト（５）に示したように、サブクラスAddBuilderにおいて、取得メソッドbuild(Object resolved)を、リフレクションを使用して実装することで、プログラムリスト（４）に示したサブクラスSetbuilderでは対応できない、paintersのような配列型のフィールドを有する、種々の被構築オブジェクトの初期化が可能となる。

【００８５】なお、プログラムリスト（５）におけるfieldNameおよびmethodNameも、プログラムリスト（４）における場合と同様に、例えば、コンストラクタによって与えるようにすることができる。

【００８６】次に、登録クラスについて説明する。

【００８７】登録クラスは、取得オブジェクトが取得した被構築オブジェクトを登録するためのクラスで、図９に示すように、被構築オブジェクトresolvedを、それに名前（文字列や数字など）nameを付して登録する登録メソッドbind(String name,Object resolved)と、名前nameから、登録した被構築オブジェクトを探し出す検索メソッドlookup(String name)を有する。従って、登録クラスのインスタンスである登録オブジェクトにおいて、ある取得オブジェクトが取得した被構築オブジェクトを、登録メソッドbind(String name,Object resolved)によって登録しておくことにより、その被構築オブジェクトの参照が再度必要となったときには、検索メソッドlookup(String name)によって、その被構築オブジェクトを探し出すことができる。

19

【００８８】具体的には、例えば、図１０に示すように、３つの被構築オブジェクトＡ乃至Ｃから構成（構築）される構築物（アプリケーションプログラムや、マルチメディアコンテンツ、その他のデータ）がある場合において、被構築オブジェクトＡのフィールドfieldAを、被構築オブジェクトＢの参照によって初期化し、その被構築オブジェクトＢのフィールドfieldBを、被構築オブジェクトＣの参照によって初期化し、さらに、その被構築オブジェクトＣのフィールドfieldCを、被構築オブジェクトＢの参照によって初期化するときには、被構築オブジェクトＢの参照は、被構築オブジェクトＡの初期化を行った後、被構築オブジェクトＣの初期化を行うにも必要となる。

【００８９】そこで、このような場合には、被構築オブジェクトＡの初期化のために、被構築オブジェクトＢを取得したときに、登録オブジェクトにおいて、登録メソッドbind(String name,Object resolved)によって、被構築オブジェクトＢを登録（bind）しておく。そして、被構築オブジェクトＣの初期化の際に、登録オブジェクトにおいて、検索メソッドlookup(String name)によって、被構築オブジェクトＢを探し出す（look up）ことで、その参照によって、被構築オブジェクトＣの初期化を行うことが可能となる。

【００９０】次に、図１１のフローチャートを参照して、ユーザ端末ＵＴ上において、構築プログラムが実行されることにより行われる構築処理について、さらに説明する。

【００９１】構築処理では、まず最初に、ステップＳ１において、取得クラス、構築クラス、または登録クラスのインスタンスである、所定の構築物を構成する被構築オブジェクトを取得するための取得オブジェクト、それらの参照関係を構築するための構築オブジェクト、または再度参照される被構築オブジェクトを登録するための登録オブジェクトが、それぞれ生成され（ユーザ端末ＵＴの内蔵するメモリ上に展開され）、ステップＳ２に進む。ステップＳ２では、ステップＳ１で生成された取得オブジェクトおよび構築オブジェクト、さらには、必要に応じて、登録オブジェクトによって、所定の構築物を構成する被構築オブジェクトが取得され、その参照関係が構築される。これにより、所定の構築物が構成され、処理を終了する。

【００９２】次に、所定の構築物として、例えば、図１２に示すような参照関係のある、８つの被構築オブジェクトｏｂｊ１乃至ｏｂｊ８からなるアプリケーションプログラムまたはマルチメディアコンテンツその他のデータが構成される場合を例に、構築処理につき、さらに説明する。なお、図１２において、被構築オブジェクトを表す四角形の下の段には、その被構築オブジェクトが有するフィールド（フィールド名）を図示してある。

【００９３】図１２では、構築物であるアプリケーションプログラムまたはマルチメディアコンテンツその他のデータが、被構築オブジェクトｏｂｊ１乃至ｏｂｊ８から構成されるため、まず、その８つの被構築オブジェクトｏｂｊ１乃至ｏｂｊ８を得るための取得オブジェクト（取得クラスのインスタンス）が必要となる。いま、この８つの被構築オブジェクトｏｂｊ１乃至ｏｂｊ８を取得するための取得オブジェクトを、それぞれGetter1, Getter2, Getter3-1, Getter4, Getter5, Getter6, Getter7, Getter8とする。さらに、図１２では、被構築オブジェクトｏｂｊ１によって参照される被構築オブジェクトｏｂｊ３が、被構築オブジェクトｏｂｊ３よりも後に取得される被構築オブジェクトｏｂｊ８によっても参照される。従って、被構築オブジェクトｏｂｊ８を初期化するときに必要となる被構築オブジェクトｏｂｊ３を取得する取得オブジェクトが必要であり、これを、Getter3-2とする。なお、Getter3-2は、登録オブジェクト（登録クラスのインスタンス）が有する検索メソッドlookup(String name)によって、被構築オブジェクトｏｂｊ３を探し出す（被構築オブジェクトｏｂｊ３の参照を取得する）ようになされている。

【００９４】また、図１２の実施の形態では、被構築オブジェクトｏｂｊ１がフィールドpPart11およびpPart12を、被構築オブジェクトｏｂｊ２がフィールドPpart21を、被構築オブジェクトｏｂｊ３がフィールドpPart31を、被構築オブジェクトｏｂｊ４がフィールドpPart41およびpPart42を、被構築オブジェクトｏｂｊ６がフィールドpPart61およびpPart62を、被構築オブジェクトｏｂｊ７がフィールドpPart71を、被構築オブジェクトｏｂｊ８がフィールドpPart81を、それぞれ有している。

【００９５】そして、図１２の実施の形態では、被構築オブジェクトｏｂｊ１のフィールドpPart11, pPart12は、被構築オブジェクトｏｂｊ３，ｏｂｊ６それぞれの参照によって、被構築オブジェクトｏｂｊ２のフィールドPpart21は、被構築オブジェクトｏｂｊ７の参照によって、被構築オブジェクトｏｂｊ４のフィールドpPart41, pPart42は、被構築オブジェクトｏｂｊ５，ｏｂｊ８それぞれの参照によって、被構築オブジェクトｏｂｊ６のフィールドpPart61, pPart62は、被構築オブジェクトｏｂｊ４，ｏｂｊ２それぞれの参照によって、被構築オブジェクトｏｂｊ８のフィールドpPart81は、被構築オブジェクトｏｂｊ３の参照によって、それぞれ初期化する必要がある。

【００９６】従って、初期化すべきフィールドとしては、pPart11, pPart12, pPart21, pPart41, pPart42, pPart61, pPart62, pPart81があり、これらを初期化する構築オブジェクトを、それぞれBuilder1-1, Builder1-2, Builder2, Builder4-1, Builder4-2, Builder6-1, Builder6-2, Builder8とすると、ここでは、この８つの構築オブジェクトが必要となる。

【００９７】なお、図１２では、被構築オブジェクトｏ

ｂｊ３が、フィールドpPart31を有し、また、被構築オブジェクトｏｂｊ７が、フィールドpPart71を有しているが、これらは、他の被構築オブジェクトの参照によって初期化されるようにはなされていないから、その初期化のための構築オブジェクトは必要ない。

【００９８】以上から、図１２の構築物を構成するには、図１３に示すような９の取得オブジェクト（取得クラスのインスタンス）と、８の構築オブジェクト（構築クラスのインスタンス）が必要となる。

【００９９】そして、これらの９の取得オブジェクトと、８つの構築オブジェクトとの関係は、図１４に示すようになっており、これらのオブジェクトにおいてメソッド（取得メソッドgetObject()、構築メソッドbuild(Object resolved)）が実行されることにより、図１２の構築物を構成（構築）する構築処理が行われる。

【０１００】即ち、まず最初に、取得オブジェクトGetter1によって（取得オブジェクトGetter1が有する取得メソッドgetObject()がコールされることによって）、被構築オブジェクトｏｂｊ１が取得される。取得オブジェクトGetter1は、被構築オブジェクトｏｂｊ１が有するフィールドpPart11, pPart12それぞれを初期化する構築オブジェクトBuilder1-1, Builder1-2を有しており（Builder型の配列builders[]の要素として、Builder1-1, Builder1-2の２つを有しており）、それぞれによって初期化が行われる。

【０１０１】即ち、まず、取得オブジェクトGetter1では、それが有するBuilder型の配列builders[]の２つの要素Builder1-1, Builder1-2のうちの１番目のBuilder1-1における構築メソッドbuild(Object resolved)がコールされる。構築オブジェクトBuilder1-1は、フィールドpPart11を被構築オブジェクトｏｂｊ３の参照によって初期化するために、その被構築オブジェクトｏｂｊ３を取得する取得オブジェクトGetter3-1を有しており、取得オブジェクトGetter3-1は、被構築オブジェクトｏｂｊ３を取得する（Builder1-1における構築メソッドbuild(Object resolved)において、取得オブジェクトGetter3-1が有する取得メソッドgetObject()がコールされることにより、被構築オブジェクトｏｂｊ３が取得される）。

【０１０２】被構築オブジェクトｏｂｊ３は初期化する必要がないため、取得オブジェクトGetter3-1は、その初期化のための構築オブジェクトを有しておらず（取得オブジェクトGetter3-1が有する、Builder型の配列builders[]の要素がnullになっており）、これにより、取得オブジェクトGetter3-1における取得メソッドgetObject()の処理が完結し、リターンする。

【０１０３】このとき、取得オブジェクトGetter3-1における取得メソッドgetObject()は、取得した被構築オブジェクトｏｂｊ３の参照を、戻り値として、構築オブジェクトBuilder1-1に返し、構築オブジェクトBuilder1-1は、その被構築オブジェクトｏｂｊ３の参照によって、被構築オブジェクトｏｂｊ１のフィールドpPart11を初期化する。そして、フィールドpPart11の初期化が終了すると、構築オブジェクトBuilder1-1における構築メソッドbuild(Object resolved)の処理が完結し、リターンする。

【０１０４】構築オブジェクトBuilder1-1における構築メソッドbuild(Object resolved)が完結すると、取得オブジェクトGetter1では、それが有するBuilder型の配列builders[]の２つの要素Builder1-1, Builder1-2のうちの２番目のBuilder1-2における構築メソッドbuild(Object resolved)がコールされる。

【０１０５】構築オブジェクトBuilder1-2は、フィールドpPart12を被構築オブジェクトｏｂｊ６の参照によって初期化するために、その被構築オブジェクトｏｂｊ６を取得する取得オブジェクトGetter6を有しており、取得オブジェクトGetter6は、被構築オブジェクトｏｂｊ６を取得する（Builder1-2における構築メソッドbuild(Object resolved)において、取得オブジェクトGetter6が有する取得メソッドgetObject()がコールされることにより、被構築オブジェクトｏｂｊ６が取得される）。

【０１０６】取得オブジェクトGetter6は、被構築オブジェクトｏｂｊ６が有するフィールドpPart61, pPart62それぞれを初期化する構築オブジェクトBuilder6-1, Builder6-2を有しており（Builder型の配列builders[]の要素として、Builder1-1, Builder1-2の２つを有しており）、それぞれによって初期化が行われる。

【０１０７】即ち、まず、取得オブジェクトGetter6では、それが有するBuilder型の配列builders[]の２つの要素Builder6-1, Builder6-2のうちの１番目のBuilder6-1における構築メソッドbuild(Object resolved)がコールされる。構築オブジェクトBuilder6-1は、フィールドpPart61を被構築オブジェクトｏｂｊ４の参照によって初期化するために、その被構築オブジェクトｏｂｊ４を取得する取得オブジェクトGetter4を有しており、取得オブジェクトGetter4は、被構築オブジェクトｏｂｊ４を取得する（Builder6-1における構築メソッドbuild(Object resolved)において、取得オブジェクトGetter4が有する取得メソッドgetObject()がコールされることにより、被構築オブジェクトｏｂｊ４が取得される）。

【０１０８】取得オブジェクトGetter4は、被構築オブジェクトｏｂｊ４が有するフィールドpPart41, pPart42それぞれを初期化する構築オブジェクトBuilder4-1, Builder4-2を有しており（Builder型の配列builders[]の要素として、Builder4-1, Builder4-2の２つを有しており）、それぞれによって初期化が行われる。

【０１０９】即ち、まず、取得オブジェクトGetter4では、それが有するBuilder型の配列builders[]の２つの要素Builder4-1, Builder4-2のうちの１番目のBuilder4-1における構築メソッドbuild(Object resolved)がコー

ルされる。構築オブジェクトBuilder4-1は、フィールドpPart41を被構築オブジェクトｏｂｊ５の参照によって初期化するために、その被構築オブジェクトｏｂｊ５を取得する取得オブジェクトGetter5を有しており、取得オブジェクトGetter5は、被構築オブジェクトｏｂｊ５を取得する。

【０１１０】被構築オブジェクトｏｂｊ５は初期化する必要がないため、取得オブジェクトGetter5は、その初期化のための構築オブジェクトを有しておらず（取得オブジェクトGetter5が有する、Builder型の配列builders[]の要素がnullになっており）、これにより、取得オブジェクトGetter5における取得メソッドgetObject()の処理が完結し、リターンする。

【０１１１】このとき、取得オブジェクトGetter5における取得メソッドgetObject()は、取得した被構築オブジェクトｏｂｊ５の参照を、戻り値として、構築オブジェクトBuilder4-1に返し、構築オブジェクトBuilder4-1は、その被構築オブジェクトｏｂｊ５の参照によって、被構築オブジェクトｏｂｊ４のフィールドpPart41を初期化する。そして、フィールドpPart41の初期化が終了すると、構築オブジェクトBuilder4-1における構築メソッドbuild(Object resolved)の処理が完結し、リターンする。

【０１１２】構築オブジェクトBuilder4-1における構築メソッドbuild(Object resolved)が完結すると、取得オブジェクトGetter4では、それが有するBuilder型の配列builders[]の２つの要素Builder4-1, Builder4-2のうちの２番目のBuilder4-2における構築メソッドbuild(Object resolved)がコールされる。

【０１１３】構築オブジェクトBuilder4-2は、フィールドpPart42を被構築オブジェクトｏｂｊ８の参照によって初期化するために、その被構築オブジェクトｏｂｊ８を取得する取得オブジェクトGetter8を有しており、取得オブジェクトGetter8は、被構築オブジェクトｏｂｊ８を取得する。

【０１１４】取得オブジェクトGetter8は、被構築オブジェクトｏｂｊ８が有するフィールドpPart81を初期化する構築オブジェクトBuilder8を有しており（Builder型の配列builders[]の要素として、Builder8の１つだけを有しており）、それによって初期化が行われる。

【０１１５】即ち、取得オブジェクトGetter8では、それが有するBuilder型の配列builders[]の要素Builder8における構築メソッドbuild(Object resolved)がコールされる。構築オブジェクトBuilder8は、フィールドpPart81を被構築オブジェクトｏｂｊ３の参照によって初期化するために、その被構築オブジェクトｏｂｊ３を取得する取得オブジェクトGetter3-2を有しており、取得オブジェクトGetter3-2は、被構築オブジェクトｏｂｊ３を取得する（Builder8における構築メソッドbuild(Object resolved)において、取得オブジェクトGetter3-2が有する取得メソッドgetObject()がコールされることにより、被構築オブジェクトｏｂｊ３が取得される）。

【０１１６】ここで、被構築オブジェクトｏｂｊ１によって参照される被構築オブジェクトｏｂｊ３が、被構築オブジェクトｏｂｊ３よりも後に取得される被構築オブジェクトｏｂｊ８によっても参照されることは、あらかじめ分かっている。このため、取得オブジェクトGetter3-1が被構築オブジェクトｏｂｊ３を取得すると、登録オブジェクトの登録メソッドbind(String name,Object resolved)がコールされ、取得オブジェクトGetter3-1が取得した被構築オブジェクトｏｂｊ３が登録されるようになされている。そして、取得オブジェクトGetter3-2は、その登録オブジェクトの検索メソッドlookup(String name)をコールすることで、既に取得されている被構築オブジェクトｏｂｊ３の参照を取得するようになされている。

【０１１７】既に取得されている被構築オブジェクトｏｂｊ３は初期化する必要がないため（図１２の実施の形態では、被構築オブジェクトｏｂｊ３は、最初から、初期化すべきフィールドを有しないが、仮に有していたとしても、その初期化は、取得オブジェクトGetter3-1が有する構築オブジェクトによって行われるべきものであるから、ここでは行う必要がない）、取得オブジェクトGetter3-2は、その初期化のための構築オブジェクトを有しておらず（取得オブジェクトGetter3-2が有する、Builder型の配列builders[]の要素がnullになっており）、これにより、取得オブジェクトGetter3-2における取得メソッドgetObject()の処理が完結し、リターンする。

【０１１８】このとき、取得オブジェクトGetter3-2における取得メソッドgetObject()は、取得した被構築オブジェクトｏｂｊ３の参照を、戻り値として、構築オブジェクトBuilder8に返し、構築オブジェクトBuilder8は、その被構築オブジェクトｏｂｊ３の参照によって、被構築オブジェクトｏｂｊ８のフィールドpPart81を初期化する。そして、フィールドpPart81の初期化が終了すると、構築オブジェクトBuilder8における構築メソッドbuild(Object resolved)の処理が完結し、リターンする。

【０１１９】構築オブジェクトBuilder8における構築メソッドbuild(Object resolved)が完結すると、取得オブジェクトGetter8における取得メソッドgetObject()が完結し、それが取得した被構築オブジェクトｏｂｊ８の参照が、戻り値として、構築オブジェクトBuilder4-2に返される。構築オブジェクトBuilder4-2は、この戻り値である被構築オブジェクトｏｂｊ８の参照によって、被構築オブジェクトｏｂｊ４のフィールドpPart42を初期化し、これにより、構築オブジェクトBuilder4-2における構築メソッドbuild(Object resolved)の処理が完結し、リターンする。

【０１２０】構築オブジェクトBuilder4-2における構築メソッドbuild(Object resolved)が完結すると、取得オブジェクトGetter4における取得メソッドgetObject()が完結し、それが取得した被構築オブジェクトｏｂｊ４の参照が、戻り値として、構築オブジェクトBuilder6-1に返される。構築オブジェクトBuilder6-1は、この戻り値である被構築オブジェクトｏｂｊ４の参照によって、被構築オブジェクトｏｂｊ６のフィールドpPart61を初期化し、これにより、構築オブジェクトBuilder6-1における構築メソッドbuild(Object resolved)の処理が完結し、リターンする。

【０１２１】構築オブジェクトBuilder6-1における構築メソッドbuild(Object resolved)が完結すると、取得オブジェクトGetter6では、それが有するBuilder型の配列builders[]の２つの要素Builder6-1, Builder6-2のうちの２番目のBuilder6-2における構築メソッドbuild(Object resolved)がコールされる。

【０１２２】構築オブジェクトBuilder6-2は、フィールドpPart62を被構築オブジェクトｏｂｊ２の参照によって初期化するために、その被構築オブジェクトｏｂｊ２を取得する取得オブジェクトGetter2を有しており、取得オブジェクトGetter2は、被構築オブジェクトｏｂｊ２を取得する。

【０１２３】さらに、取得オブジェクトGetter2は、被構築オブジェクトｏｂｊ２が有するフィールドpPart21を初期化する構築オブジェクトBuilder2を有しており、それによって初期化が行われる。

【０１２４】即ち、取得オブジェクトGetter2では、それが有するBuilder型の配列builders[]の要素Builder2における構築メソッドbuild(Object resolved)がコールされる。構築オブジェクトBuilder2は、フィールドpPart21を被構築オブジェクトｏｂｊ７の参照によって初期化するために、その被構築オブジェクトｏｂｊ７を取得する取得オブジェクトGetter7を有しており、取得オブジェクトGetter7は、被構築オブジェクトｏｂｊ７を取得する。

【０１２５】被構築オブジェクトｏｂｊ７は初期化する必要がないため、取得オブジェクトGetter7は、その初期化のための構築オブジェクトを有しておらず（取得オブジェクトGetter7が有する、Builder型の配列builders[]の要素がnullになっており）、これにより、取得オブジェクトGetter7における取得メソッドgetObject()の処理が完結し、リターンする。

【０１２６】このとき、取得オブジェクトGetter7における取得メソッドgetObject()は、取得した被構築オブジェクトｏｂｊ７の参照を、戻り値として、構築オブジェクトBuilder2に返し、構築オブジェクトBuilder2は、その被構築オブジェクトｏｂｊ７の参照によって、被構築オブジェクトｏｂｊ２のフィールドpPart21を初期化する。そして、フィールドpPart21の初期化が終了すると、構築オブジェクトBuilder2における構築メソッドbuild(Object resolved)の処理が完結し、リターンする。

【０１２７】構築オブジェクトBuilder2における構築メソッドbuild(Object resolved)が完結すると、取得オブジェクトGetter2における取得メソッドgetObject()が完結し、それが取得した被構築オブジェクトｏｂｊ２の参照が、戻り値として、構築オブジェクトBuilder6-2に返される。構築オブジェクトBuilder6-2は、この戻り値である被構築オブジェクトｏｂｊ２の参照によって、被構築オブジェクトｏｂｊ６のフィールドpPart62を初期化し、これにより、構築オブジェクトBuilder6-2における構築メソッドbuild(Object resolved)の処理が完結し、リターンする。

【０１２８】構築オブジェクトBuilder6-2における構築メソッドbuild(Object resolved)が完結すると、取得オブジェクトGetter6における取得メソッドgetObject()が完結し、それが取得した被構築オブジェクトｏｂｊ６の参照が、戻り値として、構築オブジェクトBuilder1-2に返される。構築オブジェクトBuilder1-2は、この戻り値である被構築オブジェクトｏｂｊ６の参照によって、被構築オブジェクトｏｂｊ１のフィールドpPart12を初期化し、これにより、構築オブジェクトBuilder1-2における構築メソッドbuild(Object resolved)の処理が完結し、リターンする。

【０１２９】構築オブジェクトBuilder1-2における構築メソッドbuild(Object resolved)が完結すると、取得オブジェクトGetter1における取得メソッドgetObject()が完結し、これにより、図１２に示した構築物が構成される。

【０１３０】以上のように、アプリケーションプログラムを構成する複数の被構築オブジェクトや、アプリケーションプログラムが利用するマルチメディアコンテンツその他のデータを構成する複数の被構築オブジェクトを、取得オブジェクトによって取得し、さらに、それらの被構築オブジェクトの参照関係を、構築オブジェクトによって構築するようにしたので、被構築オブジェクトを、例えば、ネットワーク上に分散させておき、必要なときに、必要な被構築オブジェクトだけを集めて、動的にリンクさせ（参照関係を構築し）、アプリケーションプログラムやマルチメディアコンテンツなどを構成することが可能となる。

【０１３１】なお、被構築オブジェクトには、それとの新たな参照関係を構築する構築オブジェクトを所有させることが可能である。即ち、被構築オブジェクトのクラスは、例えば、次のように定義することが可能である。

【０１３２】

```
class ResolvedD extends Object{
    private int fieldA;
```

27　　28

```
    private Builder builder;
    public void setFieldA(int f){
            fieldA=f;
    }
    public void setBuilder(Builder f){
            builder=f;
    }
    public void rebuild(){
            builder.build(this);
    }
}
```

・・・（6）

【０１３３】上述のようなクラスResolvedDの被構築オブジェクトでは、フィールドfieldAが、上述した場合と同様にして初期化される他、フィールドbuilderが、ある構築オブジェクトによって初期化されるが、そのように初期化されたbuilderにおける構築メソッドbuild(Object resolved)（プログラムリスト（６）では、build(this)）をコールすることで（プログラムリスト（６）では、メソッドrebuild()をコールすることで）、構築物の参照関係を、プログラム実行中に、動的に構成することが可能となる。即ち、これにより、例えば、構築物を構成した後、その構築物を構成する被構築オブジェクトの削除や、他の被構築オブジェクトとの交換、新たな被構築オブジェクトの追加などを行うことが可能となる。

【０１３４】なお、本実施の形態では、被構築オブジェクトが、ネットワーク上に分散している場合、即ち、場所的に分散している場合を例に説明したが、本発明は、構築オブジェクトが時間的に分散している場合にも適用可能である。さらに、本発明は、被構築オブジェクトが分散している場合だけでなく、ある１カ所（例えば、図２におけるＳＴＢ４のハードディスク１７など）に集中している場合にも適用可能である。

【０１３５】また、本実施の形態では、構築プログラムが、あらかじめ、ＳＴＢ４のハードディスク１７に記憶されているものとしたが、構築プログラムは、必要なときに、必要な構築物を構成するためのものを、インターネット６から入手するようにすることが可能である。この場合、ＥＰＧそのものを受信する場合に比較して、データの受信量が少なくてすむようになる。

【０１３６】さらに、本実施の形態では、本発明を、ＥＰＧの構築に適用した場合について説明したが、本発明は、ＥＰＧ以外の、アプリケーションプログラムやマルチメディアコンテンツその他のデータの構築にも適用可能である。

【０１３７】また、本実施の形態では、構築プログラムや、被構築オブジェクトを、Java言語で記述するようにしたが、本発明は、その他のコンピュータ言語にも適用可能である。但し、例えば、Javaや、Ｃ＋＋、smalltalkなどの、いわゆるオブジェクト指向言語を用いる方が好ましい。

【０１３８】さらに、本実施の形態では、取得オブジェクト、構築オブジェクト、および登録オブジェクトの３つのオブジェクトを用いて、構築物を構成するようにしたが、例えば、取得オブジェクトと構築オブジェクトだけを用いるようにし、構築オブジェクトには、登録オブジェクトが行う処理を行わせるようにすることが可能である。

【０１３９】

【発明の効果】請求項１に記載の情報処理装置および請求項９に記載の情報処理方法、並びに請求項１０に記載の記録媒体によれば、取得オブジェクトにおいて、取得メソッドによって被構築オブジェクトが取得され、その取得オブジェクトが所有する構築オブジェクトにおいて、構築メソッドによって、取得オブジェクトが取得した被構築オブジェクトが初期化される。従って、例えば、被構築オブジェクトが分散していても、その被構築オブジェクトを集めて、アプリケーションプログラムやマルチメディアコンテンツを構成（構築）することが可能となる。

【図面の簡単な説明】

【図１】本発明を適用した放送システムの一実施の形態の構成例を示す図である。

【図２】図１のＳＴＢ４の構成例を示すブロック図である。

【図３】ＥＰＧを示す図である。

【図４】図３のＥＰＧを構成する各部品がネットワーク上に分散している様子を示す図である。

【図５】取得クラスと構築クラスとの関係を示す図である。

【図６】図５を説明するための図である。

【図７】取得クラスのサブクラスを示す図である。

【図８】構築クラスのサブクラスを示す図である。

【図９】登録クラスを示す図である。

【図１０】登録クラスの役割を説明するための図である。

【図１１】構築処理を説明するためのフローチャートである。

29

【図１２】構築処理によって構築される構築物を構成する被構築オブジェクトどうしの参照関係を示す図である。

【図１３】図１２の構築物を構成するのに必要な取得オブジェクトと構築オブジェクトを示す図である。

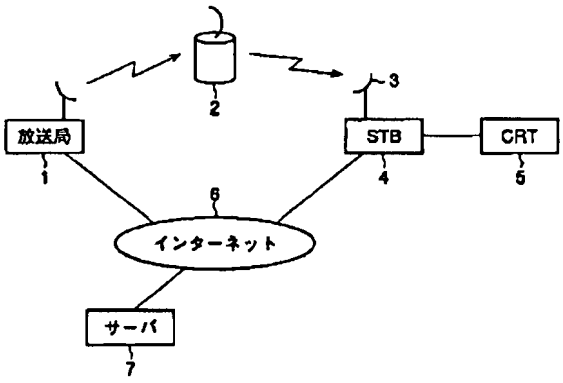
【図１４】図１２の構築物を構成するのに必要な取得オブジェクトと構築オブジェクトとの関係を示す図である。

30

【符号の説明】

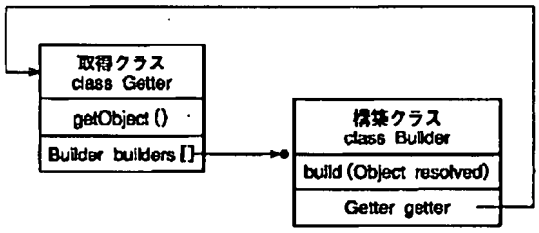
１　放送局，　２　衛星，　３　アンテナ，　４　ＳＴＢ，　５　ＣＲＴ，６　インターネット，　７　サーバ，　１１　通信Ｉ／Ｆ，　１２　受信処理部，　１３　ＲＯＭ，　１４　ＣＰＵ，　１５　ＲＡＭ，　１６　表示制御部，１７　ハードディスク，　１８　ＣＤ－ＲＯＭ，　１９　操作部

【図１】

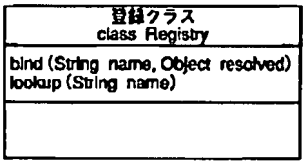


放送システム

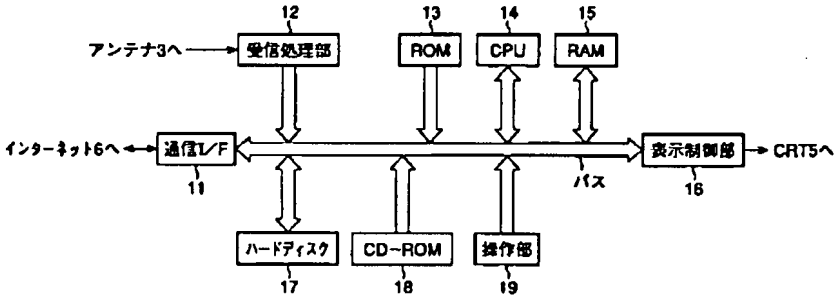
【図５】



【図９】

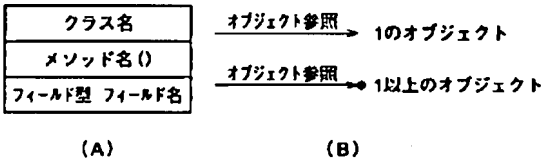


【図２】



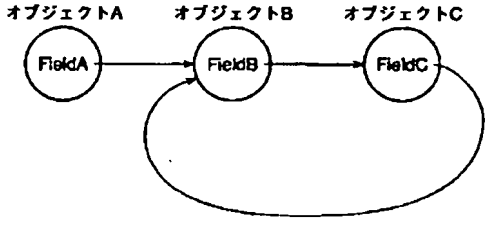
STB 4

【図６】

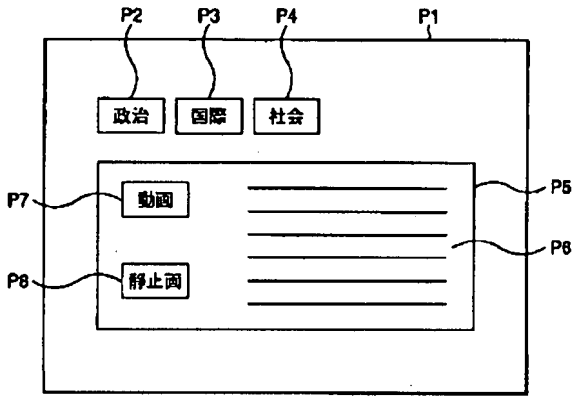


(A)　　(B)

【図１０】

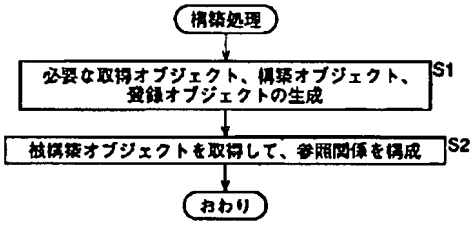


【図３】

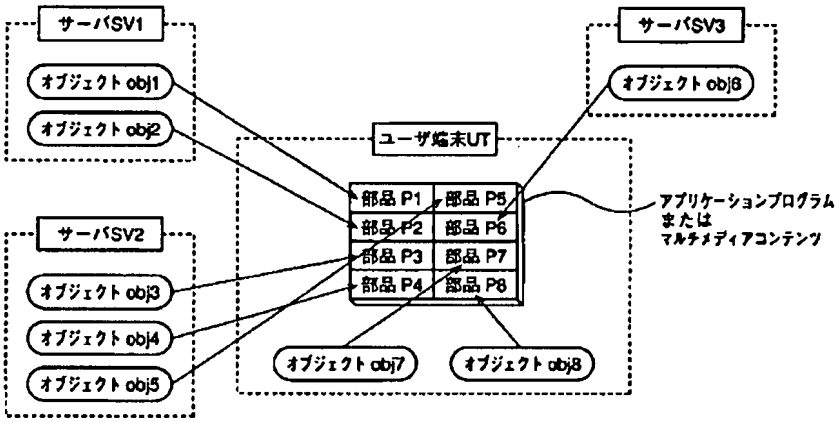


EPG

【図１１】



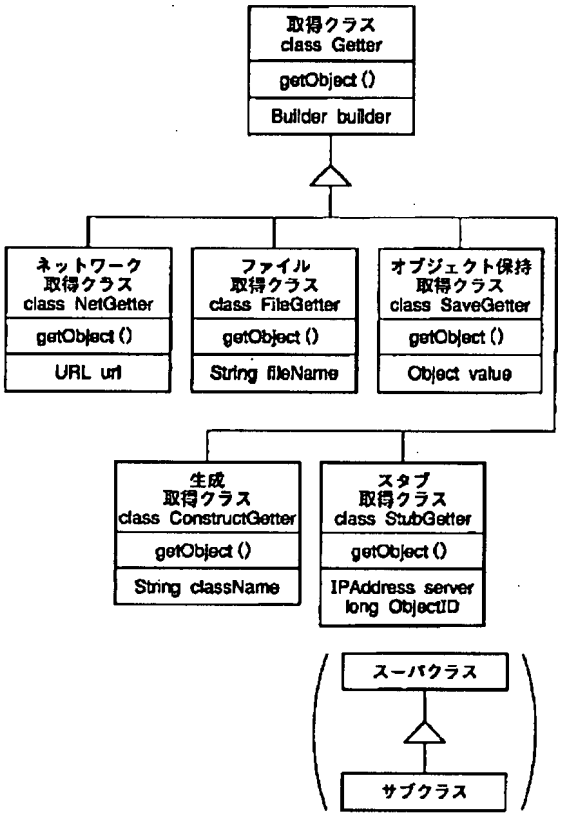
【図４】



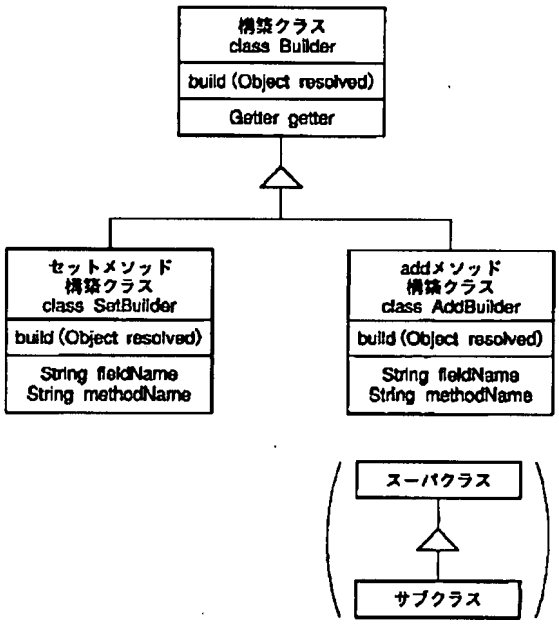
【図１３】

被構築オブジェクト	必要な取得オブジェクト	必要な構築オブジェクト
obj1	Getter1	Builder1−1 Builder1−2
obj2	Getter2	Builder2
obj3	Getter3−1 Getter3−2	
obj4	Getter4	Builder4−1 Builder4−2
obj5	Getter5	
obj6	Getter6	Builder6−1 Builder6−2
obj7	Getter7	
obj8	Getter8	Builder8

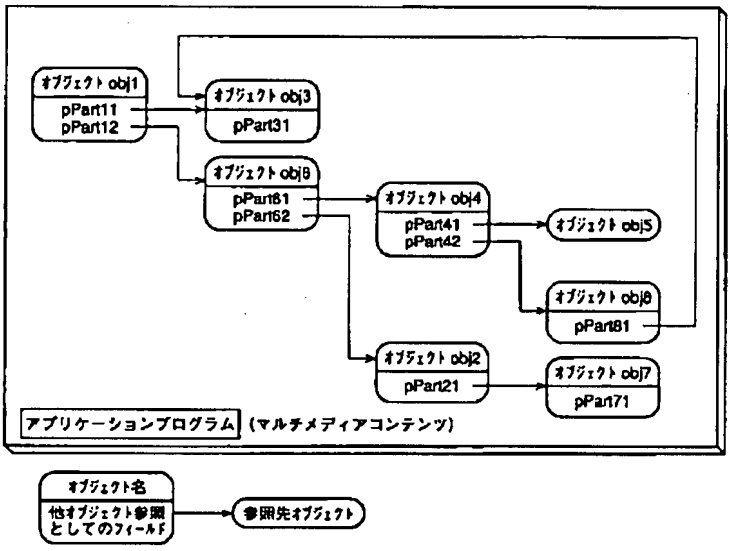
【図７】


取得クラス
class Getter
getObject ()
Builder builder
ネットワーク
取得クラス
class NetGetter
getObject ()
URL url
ファイル
取得クラス
class FileGetter
getObject ()
String fileName
オブジェクト保持
取得クラス
class SaveGetter
getObject ()
Object value
生成
取得クラス
class ConstructGetter
getObject ()
String className
スタブ
取得クラス
class StubGetter
getObject ()
IPAddress server
long ObjectID
スーパクラス
サブクラス

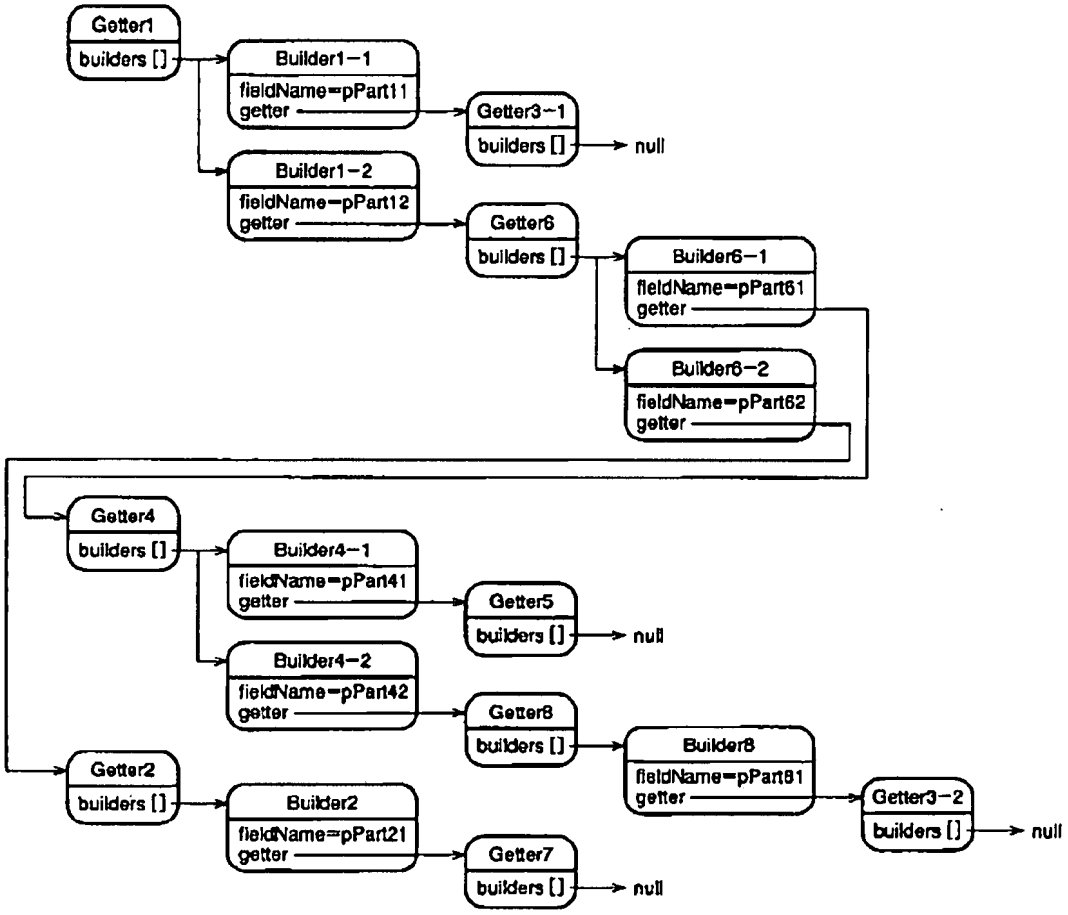
【図８】


構築クラス
class Builder
build (Object resolved)
Getter getter
セットメソッド
構築クラス
class SetBuilder
build (Object resolved)
String fieldName
String methodName
addメソッド
構築クラス
class AddBuilder
build (Object resolved)
String fieldName
String methodName
スーパクラス
サブクラス

【図１２】


オブジェクト obj1
pPart11
pPart12
オブジェクト obj3
pPart31
オブジェクト obj9
pPart91
pPart92
オブジェクト obj4
pPart41
pPart42
オブジェクト obj5
オブジェクト obj8
pPart81
オブジェクト obj2
pPart21
オブジェクト obj7
pPart71
アプリケーションプログラム (マルチメディアコンテンツ)
オブジェクト名
他オブジェクト参照
としてのフィールド
参照先オブジェクト

【図１４】


Getter1
builders []
Builder1−1
fieldName=pPart11
getter
Getter3−1
builders []
null
Builder1−2
fieldName=pPart12
getter
Getter6
builders []
Builder6−1
fieldName=pPart61
getter
Builder6−2
fieldName=pPart62
getter
Getter4
builders []
Builder4−1
fieldName=pPart41
getter
Getter5
builders []
null
Builder4−2
fieldName=pPart42
getter
Getter8
builders []
Builder8
fieldName=pPart81
getter
Getter3−2
builders []
null
Getter2
builders []
Builder2
fieldName=pPart21
getter
Getter7
builders []
null

JAPANESE PATENT (JA)

PUBLICATION

Abstract:

Disclosed are a serializing structure that MFC provides and methods for serializing the various data structure, specifically, as simple serializing methods, a method for calling a serializing ()method having attributions of person's name and birth date, month and year, and serializing member's variable values as they are in the class, responding to save/load; a serializing method of value-based collection that defines Serialize Element() of helper function of MFC templet and serializes collection of n persons starting from designated name; a serializing method comprising, entering identification code of content to be described when saving, preparing empty object corresponding to the identification code when loading, defining derivative class X of Cobject having member of virtual function and preparing Cruntime Class corresponding to the identification code per class using DECLARE-SERIAL/IMPLEMENT-SERIAL macro; a complicated elaborate serialization using CArchive; and a serialization by reference-based collection using pointer to element.

Receipt date: 19970919
Title: Object directional
Subtitle: Serializing Methods
Material type: Technical Magazine
Journal
Journal Title: DDJ DDJ
Publisher: SHE & I Kabusiki kaisha
Publish date: 19970601
Volume 6, Edition 7
Page: 120 - 132